Arquitetura Aplicação Gestão Estoque

A aplicação foi desenvolvida utilizando o C# como linguagem de programação, e as tecnologias escolhidas para as camadas foram as seguintes:

O Front-End Foi construído em Angular 8, utilizando o conceito de Single-Page-Application, criando componentes reutilizáveis pela aplicação. A escolha se deu pela familiaridade com o Angular, bem como a sua maturidade, encapsulamento e facilidade de estilização. Com uma camada front-end em Angular o front fica desacoplado do back-end e é capaz de consumir outros serviços para estender as funcionalidades.

O Back-End utilizou o framework .net core 3.1 como plataforma, foi desenvolvida uma web-api *restful*, foi utilizado também o Entity Framework para fazer o mapeamento objeto-entidade, a abordagem utilizada é a *code-first*, onde as entidades são definidas em classes c# e o entity cuida da criação de tabelas, relacionamento, etc. Esta abordagem possui a facilidade de alterar o sistema de gestão de banco de dados com pouco impacto na aplicação. O .net core foi escolhido por ser um framework moderno, de boa performance, que permite a configuração das funcionalidades a serem utilizadas pela aplicação, com ótimo suporte da Microsoft, facilidade de implantação em contêineres, nuvem ou on-premise. A versão 3.1 é a mais recente e a que contará com o suporte a longo prazo da Microsoft (LTS). A construção da api restful permite o desacoplamento da camada de negócios (.net) da camada de apresentação (Angular), permitindo o reuso da api em outras aplicações e a decomposição em micro-serviços.

O banco de dados escolhido foi o Sqlite pela simplicidade de desenvolvimento da prova de conceito, que não necessitava de recursos avançados de banco de dados, como triggers, views, stored procedures, etc. Devido a abordagem code-first, caso a aplicação venha a necessitar de um BD mais robusto, a sua substituição é sem maiores problemas. O arquivo do BD se chama “banco.db” e está na raiz do projeto.

A tela de gestão de produtos foi construída a partir de componentes encapsulados, por exemplo, todas as chamadas à API estão contidas em uma classe de serviços, seguindo boas praticas do Angular. Foram criados módulos que caso o sistema necessite de extensão é simples adicionar novas entidades. Também foram utilizados conceitos de injeção de dependência tanto no front-end quando no front-end e criação de interfaces.

Para auxiliar na documentação, visibilidade, testabilidade da API, foi integrado o módulo do Swagger disponível em “/swagger/index.html” na url da aplicação.